# Evaluating Intelligence in Unmanned Ground Vehicle Teams

Sesh Commuri[1], Yushan Li[1], Dean Hougen[2], Rafael Fierro[3]

[1]School of Electrical and Computer Engineering
University of Oklahoma, Norman, OK
[2]School of Computer Science
University of Oklahoma, Norman, OK
[3]School of Electrical and Computer Engineering
Oklahoma State University, Stillwater, OK

## ABSTRACT

Evaluation of intelligence in Teams of Unmanned Ground Vehicles (UGVs) requires the development of consistent metrics and benchmarks. This is a complicated process as the implementation of the UGVs is problem and domain specific. Different performance requirements give rise to different set of metrics making the comparison of performance between two implementations difficult. In this paper, we focus on three aspects of intelligence, namely reconfiguration, adaptation and learning, and communications in UGV teams and investigate the development of metrics for measuring their performance. We also investigate the available benchmarks for intelligent systems and verify their suitability for measuring the performance of UGV teams. A hierarchical architecture called Adaptation and Learning at All levels ($AL^2$) for the UGV teams is presented. This architecture is designed to allow for a modular and hierarchical approach to implement deliberative and reactive behaviors in teams of autonomous vehicles. In this implementation, system intelligence is incorporated at all levels of the hierarchy. The performance of the proposed architecture is evaluated using the metrics identified.

**Keywords**: *Performance Metrics, Intelligent Systems, UGVs, Adaptation and Learning.*

## 1. INTRODUCTION

The high cost associated with the acquisition and deployment of mobile robots motivates the development of low cost multi-robot teams that can function cooperatively to achieve specified goals. As the performance requirements get more stringent and the application realm becomes more diverse, embedding intelligence in system becomes a critical part of the realization of the Unmanned Ground Vehicles (UGV) Teams.

The notion of "intelligence" and the requirements for such intelligent systems has been discussed in detail in [1]. For a control system, at a very minimum level, system intelligence implies the ability to sense the environment, to make the control decisions based on the task requirements and to take the necessary corrective actions. At a higher level, system intelligence may include the ability to recognize objects and events, to represent the knowledge in a world model and to plan for the future. Intelligence at its highest level provides the ability to perceive and understand, to predict outcomes based on actions, to choose wisely between actions, and to maximize the chances of success under variety of circumstances. In general, intelligence embodies the ability to learn from experience and adapt successfully to the environment. For successful implementation, however, "intelligence" has to be formalized and the required metrics for its measurement developed.

The challenges and issues in defining performance metrics for intelligent systems are discussed in [2, 3]. The analysis of the system architecture and configuration is proposed to develop a measure of "Machine Intelligent Quotient (MIQ)" in [4]. In [5], requirement specifications and system verification are used to develop a formal method to specify the performance metrics. Techniques to assign metrics to intelligent systems are also explored in [6, 7].

Designing intelligent systems is a complex task requiring the integration of a diverse set of hardware and software components. Intelligence can be formally defined as "the ability of a system to behave appropriately in an uncertain environment" where "appropriate behavior maximizes the likelihood of the system's success in achieving its goals [1]". Such an intelligent system should be able to respond to sensory feedback at every level such that goals are achieved despite perturbations and unexpected feedback. Since intelligence responds to sensory feedback at all levels, overall effectiveness requires such 'system intelligence' to be distributed in nature. Therefore, any measure of the "intelligence" must account for the "intelligence" at each level of the system. Typical components that are to be looked at are:

a. Sensors and actuators
b. Knowledge representation and world model
c. Planning and control
d. Learning and adaptation

In addition to the above, metrics are required to measure the level of system autonomy. Some of the measures

| Report Documentation Page | | *Form Approved* *OMB No. 0704-0188* |
|---|---|---|

| 1. REPORT DATE **AUG 2004** | 2. REPORT TYPE | 3. DATES COVERED **00-00-2004 to 00-00-2004** |
|---|---|---|
| 4. TITLE AND SUBTITLE **Evaluating Intelligence in Unmanned Ground Vehicle Teams** | | 5a. CONTRACT NUMBER |
| | | 5b. GRANT NUMBER |
| | | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | | 5d. PROJECT NUMBER |
| | | 5e. TASK NUMBER |
| | | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Oklahoma,School of Electrical and Computer Engineering,Norman,OK,73019** | | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES
**Proceedings of the 2004 Performance Metrics for Intelligent Systems Workshop (PerMIS -04), Gaithersburg, MD on August 24-26 2004**

14. ABSTRACT
**see report**

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **7** | |

proposed in literature measure the sensitivity of the implementation to environment and the learning algorithms that are implemented [8]. For a system comprising a single robot, some possible metrics [9] are (a) the ability to choose strategies or algorithms; (b) the ability to generate reactive and deliberative behaviors; (c) effectiveness in accomplishing goals and objectives; (d) efficiency of operations.

While the above metrics are adequate to describe high level system performance, they do not give insight into the functioning of UGV teams. In the case of UGV teams the mission complexity and uncertainty in the environment impose more stringent requirements on the "system intelligence". In this case, the design must address additional issues such as:

1) Dynamic reconfiguration of the UGV teams to meet mission requirements. This situation is typically encountered when a new team has to be formed, or when a team has to be augmented with additional resources. Dynamic modification of teams also occurs during formation control of UGVs.
2) Coordination and Cooperation between team members, and
3) Distributed real-time communications between team members and other teams.

These requirements have strong impact on the implementation of every level in the hierarchy of the system. Thus, in addition to the metrics for the overall system performance, the performance analysis requires the definition of metrics for each level of the hierarchy in the implementation.

In this paper, a hierarchical architecture called Adaptation and Learning at All Levels ($AL^2$), that allows system intelligence to be incorporated at all levels of the hierarchy is proposed. This architecture is modular, scalable and flexible. Specific requirements on the UGV teams are listed and their impact on the entities in each layer is discussed. The development of metrics is then discussed based on this implementation framework. The rest of this paper is organized as follows: Section 2 presents the $AL^2$ architecture for teams of UGVs. The metrics needed to evaluate the performance of the team are discussed in Section 3. The implementation of the proposed architecture for the development of intelligent teams of unmanned ground vehicles is then presented in Section 4.

# 2. $AL^2$ ARCHITECTURE

In this section, architecture is proposed that enables the design of complex hierarchical systems using simple components whose performance can be rigorously analyzed. This architecture, called Adaptation and Learning at all Levels ($AL^2$), allows for intelligence to be implemented at all levels of the hierarchy and for adaptation and learning occurring at different granularities throughout the system.
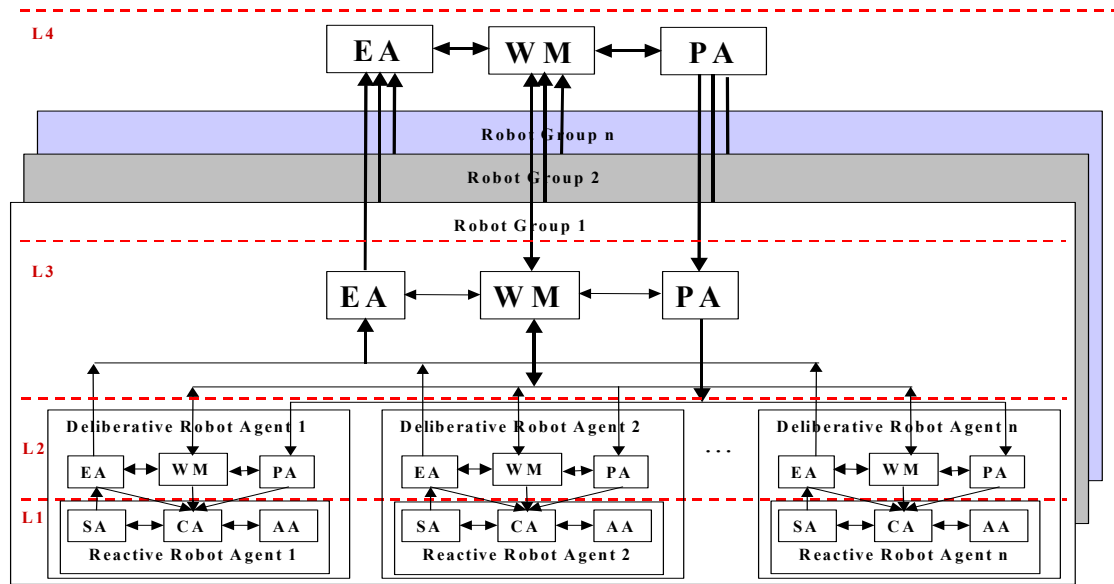


**Fig. 1 Architecture for Adaptation and Learning at All Levels ($AL^2$)**

The proposed architecture (AL$^2$) envisions one or more robotic agents working as a group. At the lowest level (L1), each robot agent has a control agent (CA), an actuator agent (AA), and a sensor agent (SA). The control agent is responsible for attaining the commanded system performance at the lowest level. It can command the sensor agent to override its output values, recalibrate its signal, as well as perform rudimentary signal processing like filtering. The AA, CA, and SA have the lowest level of autonomy. This level (L1) is characterized by stringent real-time requirements and deterministic behavior. At a very fundamental level, this design is adequate for a robotic agent to function and perform repetitive tasks in a structured environment. Note that, because of our distributed communication infrastructure, the sensor, actuator, and control sources (and the corresponding sub-agents) for a single robot agent need not be present on the same physical platform. For example, a platform lacking a camera and image processing capabilities could still perform leader-following if the leader platform had a rear-facing or omnidirectional camera (or other sensor) that could be used to sense the relative position of the follower platform.

In order to meet the requirements of fault tolerance, uncertainty in the system model and the environment, we propose a distributed architecture wherein the higher layer (L2) incorporates elements that instill higher-level intelligence in the robot. In this layer, the sensory signals from Layer L1 are processed by the Estimator Agent (EA). The output of the Estimator is then used to modify/update the local representation of the World Model (WM) and as input to the Control Agent. The distributed intelligence paradigm that is proposed means that EA can now include algorithms for fault detection, dynamic sensor reconfiguration, and sensor fusion at the level of a deliberative robot agent. The WM entity in the robot agent maintains information about the environment that is necessary for the successful tasking of the robot. Typically, this would include local map information, friend/foe classification, targets and obstacles etc. The Planning Agent (PA) utilizes the information from the local model of the world (WM) and the high-level task requirements to generate a plan that is communicated to the control agent in layer L1. The PA implements algorithms for path planning, obstacle avoidance, optimization, etc., for an individual robot. Level L2 is characterized by increased autonomy and less stringent real-time requirements.

A team of robots consists of a number of individual robot agents possibly with differing sensor/actuator suites and capabilities. The coordination between these agents is managed by the PA entity at the level of the robot group (L3). Information sharing between L2 entities is controlled by the entities in L3. This increases the security of the implementation because the L2 entities can function independently of each other, while still functioning in a coordinated manner. The primary function of the entities

in Layer 3 is to coordinate the working of the robot agents in the group. L3 handles all reassignments of tasks between different robot agents in L2. Introduction of new robot agents or sensor suites, etc., are the exclusive domain of L3. The outputs of all the EAs in layer L2 provide the input to the EA module in L3. Team-level sensor fusion amongst the different robotic agents is accomplished by the EA at L3. This EA module is used to update the world model (WM) in Layer 3. This WM also manages the information sharing among the robot agents in L2. The planning agent (PA) in this layer does the task decomposition from the mission requirements and updates the individual PAs in L2. It is to be noted that the architecture specified is independent of hardware and software implementations and individual elements in L2.

Layer 4 (L4) manages the coordination between groups of robot agents. The highest level of intelligence and autonomy and the lowest level of real-time criticality characterize L4. Dynamic reassignment of the responsibilities of each group is handled by L4.

The proposed architecture will enable the development of groups of unmanned ground vehicles that can be dynamically configured and retasked. The architecture is flexible and is not dependent on the type of controllers or algorithms implemented in any given layer.

# 3. DEVELOPMENT OF METRICS FOR SYSTEM INTELLIGENCE

In this section, the metrics required to evaluate the performance of the UGV teams with respect to reconfiguration, cooperation and the real-time communication between team entities are addressed.

## 3.1 Dynamic Reconfiguration of Hardware and Software

To meet the operational requirements for different tasks, the system should have the intelligence to implement different software and hardware configurations dynamically at the every level of the hierarchy. Plug-and-play sensors and actuators require relevant signal processing elements and software drivers to be loaded and the data made available seamlessly to the application. Fault handling on the other hand might necessitate the routing of signals dynamically through a different part of the system in order to bypass a faulted element. The capability of the system to handle these requirements can be evaluated by the following set of metrics.

### a. Is the system reconfigurable?

The rigidity of the implementation can be assessed by checking the amount of reconfigurable entities in the implementation. A typical implementation may consist of fixed hardware and software components and some

modules whose functionality can be modified at run-time either by the user or by other processes. The amount of reconfigurable resources as a ratio of the overall system resources is a measure of the reconfigurability of the system.

### b. Is the system reconfiguration static or dynamic?

Static reconfiguration requires the system to be taken off line and reconfigured before it is deployed. On the other hand, dynamic reconfiguration can take place while the system is under operation. Dynamic reconfiguration is essential when it is not feasible to take the system off-line to implement changes.

### c. Can the system be fully / partially reconfigured?

Full and partial reconfigurations are important aspects of the design of intelligent systems. Systems typically need to execute special tests at startup to verify proper system functioning. Once the startup tests are complete, the system can transition to the "run-time" mode. While it is easy to load test software to run system tests at startup, the tests that can be run are constrained by the hardware. By incorporating the ability to change the configuration of the hardware, for example by using FPGA devices, the same hardware can be used for system tests at startup and then "fully" reconfigured for run-time operations. The ability to "fully" reconfigure the hardware is also essential to the retasking of the individual robot. When the robot is retasked, sensor and actuator configurations can be selected that adapt the robot for the specified task. Since the embedded hardware can be optimized for the specific task, the overall performance can be improved without an increase in system cost.

Often, it is required to re-route the signals to accommodate for faults or add additional circuitry to handle signals from new sensors that come on-line. In such circumstances, unused portions of the FPGA can be configured to handle this requirement while the rest of the device is unaffected. Such reconfiguration, called Partial Reconfiguration, is essential to support retasking of individual robots, plug-and-play transducers, and for fault accommodation.

### 3.2 Coordination and Cooperation Between Team Members

Traditional control theory enables the design of controllers in a single mode of operation in which the task and the controlled plant are fixed. Contrary to this, in the case of UGV teams, team members usually interact with each other and with uncertain or unstructured environments. The team is to reach a goal destination, negotiate around obstacles and satisfy constraints on the formation. Thus, any measure of performance of the UGV teams has to address the ability of the teams to coordinate and cooperate with another in order to successfully accomplish the overall mission objective. The effectiveness of the implementation of UGV teams for coordination and cooperation among team entities can be measured by the following set of metrics.

### a. Is the UGV team capable of executing simple formations?

The ability of the UGV team to execute simple formations like leader-follower, straight line, and convoy are a good indication of the coordinated activity among the members of the UGV team.

### b. Is it possible for individual members in the team to seamlessly share sensory information, world models, and data?

The performance of the UGV team can be significantly improved if the sensors could be calibrated using reference data gathered by an external entity. The ability to use sensory information from other team members also extends the capability of a team in the face of sensory failures. Successful implementation of intelligent UGV teams requires the ability for each member of the team to benefit from the knowledge gained by other team members. Thus by sharing the world models and the knowledge, UGVs can demonstrate behaviors that are not programmed. Similarly, sharing of performance data between members is critical to the efficient operation of the UGV team.

### c. Can an UGV team be dynamically modified by the addition or removal of a team member?

Operational damage to an UGV or changing mission requirements often requires augmentation of an UGV team with additional resources. The ability to recognize the availability of additional resources and retask each of the UGVs in the team is a measure of the dynamic adaptability of the UGV team. This characteristic is also important in cases where an UGV team moving in a formation has to navigate around a dynamic obstacle. In this case, the team has to split into two sub-teams, maintain sub-formations while avoiding the obstacle and then rejoin in the original formation.

## 3.3 Distributed Real-Time Communications

Real-time communications are essential to support other functions within the UGV teams. The communication scheme has to be flexible and allow for communication of differing transmission rates, media, and security. The ability of the system to dynamically select channels of communication to improve performance and reduce power consumption is critical to the performance of the team. The following characteristics can be used as a measure of the performance of the communications scheme.

### a. Is there a mechanism for the intra-layer and inter-layer communications?

Teams of UGVs have to communicate and coordinate at several levels [11]. Therefore in each layer, modular implementation of the entities with appropriate communication interfaces is crucial for the successful coordination between UGVs. The communication mechanism will have to provide visibility into each entity at every level of the implementation. For example, the ability to share sensory data, world models, or plans between different UGVs in a team is essential to the implementation of intelligent UGVs. Intra-layer and inter-layer communication is essential for the intelligence of an UGV while communication between different UGVs and teams of UGVs is crucial for the implementation of intelligent teams of UGVs.

### b. Is the communication scheme used flexible?

The higher layers in the implementation of an UGV are typically characterized by abstract entities where intelligent decisions are made. The lower layers on the other hand, are characterized by real-time control modules where traditional closed loop control decisions are taken. The varied nature of communication at each level in the implementation of an UGV implies that the communication scheme employed must be flexible enough to enable the dissemination of high-level abstract information as well as the low-level real-time data.

### b. What are the communication mechanisms and the guaranteed performance?

Successful operation of the UGV teams requires a number of communication techniques. Commonly implemented ones are the ability to provide query-and-response mechanism, broadcast and periodic transmission of data between different entities in the implementation. Key properties such as bandwidth, transmission rates, protocol overhead, transmission error rates, error correcting methods etc. are to be analyzed to ensure that the communications do not become a bottleneck in the performance of the overall system.

## 4. CASE STUDY

The proposed framework is tested by implementing the L1 layer of the proposed architecture on the Xilinx Virtex-II Pro platform [13]. This platform was selected based on its capability to implement reconfigurable architectures, and the excellent development tools and product support. The Virtex-II Pro XC2VP4 has a PowerPC core, 6768 logic cells, 504 KBits BRAM, 4 3.125 Gbps RocketIO transceivers, and 3.01 Mbits configuration space.

The Xilinx Virtex-II Pro device is a user programmable gate array with embedded PowerPC processor and embedded high-speed serial transceivers. The Xilinx Virtex architecture is coarse grained and consists of a number of basic cells called configurable logic blocks (CLBs). These logic blocks are arranged in rows and columns, with each CLB consisting of four logic cells arranged in two slices. Each CLB also contains logic that implements a four-input look up tables (LUTs). Each slice contains two function generators, two storage elements, arithmetic logic gates, large multiplexers, wide function capability, fast-carry look ahead chains, and horizontal cascade chains. The function generators are configurable as four input look up tables (LUTs), sixteen bit shift registers, or as sixteen bit selective RAM memory. Each CLB also has fast interconnect and connects to a generalized routing matrix (GRM) to access general routing resources. The Virtex-II Pro has SelectIO-Ultra blocks (IOBs) that provide the interface between the package pins and the internal configurable logic. Active Interconnect Technology connects all these components together. The overall interconnection is hierarchical and is designed to support high speed designs.

The programmable elements in the Virtex-II Pro, including the routing resources, are controlled by values stored in the static memory cells. The device is configured by loading the bitstream into the internal configuration memory. These values can be reloaded to change the functions of the programmable elements. The Xilinx Virtex family of FPGAs supports both partial as well as dynamic reconfiguration. Partial reconfiguration can be achieved in one of the two ways, namely Module-based partial reconfiguration and difference-based reconfiguration. In the module-based reconfiguration, the entire module can be reconfigured. The height of the reconfigurable module is the height of the device and the module can cover one or more columns. In difference-based reconfiguration, the reconfiguration is done by making a small change in the design, and then generating a bit-stream based only on the differences in the two designs. Switching the configuration from one implementation to another is easy and very quick.

The system is designed with PPC405 processor core, SDRAM controller connected to Processor Local Bus (PLB) and general purpose Inputs-Output (GPIO) devices like Leds, Push buttons, UART and dip switches are connected to its On-chip Peripheral Bus (OPB). These are the components available on

**Fig. 2.** Implementation of PWM motor control with dynamic reconfiguration for fault accommodation.



**Fig. 3.** Neural Network based compensation of actuator nonlinearities.



**Fig. 4.** Implementation of a one-layer Neural Network in Simulink using Xilinx Toolset.

the board, so the first step is to verify the proper functionality of all these components. To do this, the processor boots up with a configuration file to test all the components. On successful completion of built in self test, the processor fetches the second configuration file to configure itself and the board, switching into operational mode. If subsequent reconfiguration of the system requires additional components, then the configuration files can be loaded dynamically into the FPGA. For example, if serial communication capability is required, then a preconfigured UART module can be loaded into the FPGA and the relevant software drivers activated. This can be achieved during the operation of the system with configuration times of the order of micro-seconds.

In the second design example, a PWM generator is implemented in the hardware to control the drive motors of the robot (Fig. 2). Timer 1 (pwmTimer) is configured to generate the PWM signal while Timer 2 (opbTimer) is configured to sense the feedback signal. If a fault is detected in the "sense" circuit during operation, then a different sensing circuit is dynamically created and the signals routed through it. The control cycle in this example was executed in real time with a sampling rate of 20 milli-secs. The time for reconfiguration was of the order of a few micro-seconds showing that dynamic fault accommodation was achieved in real time.

In the third design example, a Neural Network (NN) is implemented in the FPGA to compensate for actuator deadband (Fig.3). The controller continuously monitors the output and dynamically instantiates the neural network in the FPGA when the performance degrades significantly due to load dependent deadband in the actuator dynamics. The Neural Network is modeled and designed in Simulink using the Xilinx toolset provided by Mathworks Inc. (Fig. 4). Once the design has been successfully validated, a configuration bit stream can be generated that allows for dynamic creation of the NN module in the hardware.

## 5. CONCLUSIONS

In this paper, architecture was presented that facilitates the implementation of teams of intelligent UGVs. Three aspects of intelligence, namely reconfiguration, adaptation and learning, and communications in UGV teams were investigated and metrics for measuring their performance were proposed. The performance of the proposed architecture was evaluated using the metrics identified.

## REFERENCES

[1]. J. S. Albus. Outline for Theory of Intelligence. IEEE Trans. System, man and Cybernetics, Vol. 21, No.3, pp473-509, 1991.

[2] J.M. Evans, E.R. Messina, Performance Metrics for Intelligent Systems. . Proc. of Performance Metrics for Intelligent Systems Workshop. 2000 PerMIS Workshop, 2000.

[3] L.A. Zadeh. The Search for  Metrics of Intelligence- A Critical View. Proc. of Performance Metrics for Intelligent Systems Workshop. 2000 PerMIS Workshop, 2000.

[4] E.C. Chalfant, S. Lee. Measuring the Intelligence of Robotic Systems: An Engineering Perspective. Proceedings of International Symposium on Intelligent Systems. Gaithersburg, MD. October, 1999.

[5]. Y.Zhang, A.K. Mackworth. Formal Speification of Performance Metrics for Intelligent Systems. Proc. of Performance Metrics for Intelligent Systems Workshop. 2000 PerMIS Workshop, 2000

[6]. C. Schlenoff, L. Welsch, R. Madhavan, N. Zimmerman, Towards Measuring the Performance of Architectural Components of Autonomous Vehicular Systems. 2002 PerMIS Workshop August 13-15, 2002.

[7]. J. S. Albus. Features of Intelligence Required by Unmanned Ground Vehicles.  Proc. of Performance Metrics for Intelligent Systems Workshop. 2000 PerMIS Workshop, 2000.

[8] A.Yavnai. Metrics for System Autonomy, Part I: Metrics Definition. Proc. of Performance Metrics for Intelligent Systems Workshop. 2000 PerMIS Workshop, 2000

[9] R. Finkelstein. A Method for Evaluating the 'IQ' of Intelligent Systems. Proc. of Performance Metrics for Intelligent Systems Workshop. 2000 PerMIS Workshop, 2000.

[10] Sesh Commuri, Yushan Li, Dean Hougen, Rafael Fierro. Designing for System Intelligence. Proceedings of the 5th IFAC Symposium on the Intelligent Autonomous Vehicles, Lisbon, Portugal, July, 2004.

[11] R.Muthuraman, A. Fajebe, S.Commuri. Intelligence in Embedded Controls-A Case Study. Proceedings of the IEEE Region 5 Technical Conference, April 2004.

[12] S. Donti, and R. L. Haggard, "A survey of dynamically reconfigurable FPGA devices, " Proc. IEEE., vol. 8, pp. 422-426, 2003.

[13] Xilinx, Inc., Virtex-II Pro: Platform FPGA Handbook, UG012, v 2.0., 2002.